229-D2 36383

Calculus Computer Lab
Dr Matthew Sunderland

1. Synchronous lecture Friday 10:10–12:05
   https://zoom.us/meeting/register/tJYudO6sqjooHNegs0PYh2HgKUWkXYf7T06G
2. Online problem sets (labs) due Sundays (9 days after each lecture)
   https://bbhosted.cuny.edu
3. Written assignments due some Sundays on
   https://www.gradescope.com course code M8PW4X
4. Reading assignments due each night before lecture
   https://www.perusall.com course code SUNDERLAND-GK4L9
5. MATLAB is required. Go to https://www.mathworks.com/login?form_type=tah_portal&uri=https%3A%2F%2Fwww.mathworks.com%2Flicensecenter%2Ftotal_headcount%2F14317-60551-55097-39870-91449%3Fs_tid%3Dtah_po_start_cuny click "No account? Create one!" and use your CSI email
6. Office hours TBD
   https://zoom.us/my/mattsunderland
7. Announcements, Lecture Recordings, and Grades posted on
   https://bbhosted.cuny.edu
8. Platform for administering exams TBD,
   possibly Blackboard, Gradescope, WeBWorK, Respondus, or Proctortrack

Day 1 Homework

1. Download Zoom and create free account
2. Do Online Problem Set 1 (Lab 1) by Sunday 9/6
3. Submit Written Assignment 1 by Sunday 8/30—see last two pages of syllabus
4. Do first reading assignment (Lab 2) and make 1 comment by Thursday 9/3
5. Download and install MATLAB on your computer.
6. Do office hour survey https://forms.gle/RRf74atLQkR3kg5DA

Course Grade = Average of
- Coursework
  1. Lecture participation
  2. Online problem sets
  3. Written assignments
  4. Reading assignments
- Exam 1
- Exam 2
- Final

Lecture Recording Statement *Students who participate in this class with their camera on or use a profile image are agreeing to have their video or image recorded solely for the purpose of creating a record for students enrolled in the class to refer to, including those enrolled students who are unable to attend live. If you are unwilling to consent to have your profile or video image recorded, be sure to keep your camera off and do not use a profile image. Likewise, students who un-mute during class and participate orally are agreeing to have their voices recorded. If you are not willing to consent to have your voice recorded during class, you will need to keep your mute button activated and communicate exclusively using the "chat" feature, which allows students to type questions and comments live.*

Deadlines Add 9/1 Drop 9/15 Withdraw 11/6

clear screen by typing:
clc


```
>> b/a
>> a=a-18
>> a^b
```

In coding, "=" means "record it as"

When you make an assignment, the variable name appears in the Workspace viewer which appears in the upper left pane of the initial MATLAB window. You can view the contents of the variable by double clicking it in the workspace viewer. Otherwise, you can enter just the variable name in the command line and its contents will be displayed.

Exercise 1:

a. What is the output of the following commands:

```
>> a=3; b=4; c=5;
>> a + b/c
```

(1) Circle one:
1. 3 4/5
2. 1.6
3. 3.8
4. 7/5

Example 1:
Assigning variables can simplify matters when used wisely. For example, when evaluating

$$\frac{(2-3)-(-3)}{(-1)+2}.$$

One way is to write out the whole expression at once:

```
>> ((2-3)-(-3))/((-1)+2)
ans = 2
```

There are many parentheses that are needed to get this right. This can make finding errors in our work tough. If we use spaces (which are ignored) and intermediate names, we can reduce the chance of a typing error:

```
>> top = (2-3) - (-3);
>> bottom = (-1) + 2;
>> top/bottom
ans = 2
```

This technique makes errors much easier to find.

b.

$$\frac{a^{(c-b)}}{c-b}$$

(19) Answer: _____

c.

$$\frac{a^{3/2}}{b}$$

(20) Answer: _____

d.

$$\frac{a-b(c-a)}{c-a}$$

(21) Answer: _____

Note: A full explanation of MATLAB's priority rules is attached to this project in the **Reference Section**. You may want to review this to check your answers.

3.3.1 Reusing previous commands

You can save some typing if you learn how to reuse and edit your previously entered commands. The **Command History** window shows the commands you've typed during a MATLAB session. Double clicking on a command will paste it into the command window. The command history may also be accessed inside the command line by using either the up or down arrow to scroll through your previous commands.

Once a command is at the >> prompt, you may make changes to it. Use the left or right arrows to move around, or the mouse. Then you may insert text or delete existing text.

Finally, you may type `Enter` to execute the new command. You do not need to have the cursor at the end of the line to do this.

4 Function evaluation

MATLAB allows much more than just the basic arithmetic operations. Extra functionality is provided by functions, such as `sin`, `cos`, or `sqrt`. A function is referred to by its name and used by calling the function with its argument(s) enclosed in matching parentheses. A list of basic functions and their MATLAB equivalents is attached to the end of the project in the Reference Section.

For instance, the value of $\sqrt{15}$ is given by

```
>> sqrt(15)                    #  ans = 3.8730
```

```
>> sin(2*pi)

ans =

   -2.4493e-16
```

this is scientific notation for

-2.4493×10^{-16} (negative exponent)

this is super tiny

* when you see this, the real answer is 0 (type in zero as your answer for the exercise)

but the computer made some rounding error

d. Evaluate $\arcsin(\sin(3\pi/4))$
(29) Circle one:
1. $3\pi/4$
2. $-5\pi/4$
3. $\pi/4$

fancy word for "list of numbers"
/ sometimes also called an "array"

5 Vectors

We will often want to apply a function to many different values of x. We'd like to do this in the most convenient manner. For example, to compute the function $f(x) = x^2\cos^3(x)$ for $x = \pi/3, \pi/4$ and $\pi/6$ we can save some work by assigning a value to x and then computing:

```
>> x=pi/3; x^2 * cos(x)^3
>> x=pi/4; x^2 * cos(x)^3
>> x=pi/6; x^2 * cos(x)^3
```

(use up arrow to recall past lines)

This allows us to make a single change to x per line, instead of changing it in both places.

Although the above technique is useful, there are better ways to do this task, as the MATLAB language is written to naturally apply the same function to many different values at once. In order to do so we need to learn two things:

a. How to store more than one number into a variable (vectors)

b. How to apply a function to all the values of the vector simultaneously.

5.1 Defining vectors

We use the term vector to describe a MATLAB variable that contains lots of numbers at once. Vectors are made in MATLAB using the square brackets `[]`. (MATLAB refers to vectors as arrays, a more general concept.)

The simplest way to make a vector is to just type in the numbers you want inside of matching `[]`:

(Don't type the part with the % symbol. This is a comment to you.)

```
>> x = [1,1,2,3,5,8,13,21]      % start of Fibonacci sequence
>> x = [1 1 2 3 5 8 13 21]      % commas are optional
>> somePrimes = [2,3,5,7,11,13,17,19,23]
```

```
>> x = [pi/3, pi/4, pi/6]; x.^2 + cos(x).^3
```

Matlab will do the computation on each of the 3 numbers seperately ("element-wise")

When you raise a vector (list of numbers) to a power, must use "dots" (so write .^).

Same for .* ./

Exercise 9:
Store the number $1, 2, 3$ in a vector named `x`. Answer the following for this vector.

a. What is `x+x`?
(30) Circle one:
1. The vector `[1, 4, 9]`
2. The vector `[2, 4, 6]`
3. An error

b. What is the output of `x * x`
(31) Circle one:
1. The vector `[1, 4, 9]`
2. The vector `[2, 4, 6]`
3. An error

5.2 Arithmetic sequences

Many of the vectors of numbers we will deal with will be arithmetic sequences:

$$a, a+h, a+2h, a+3h, \ldots, a+kh = b, \qquad h > 0.$$

We think of this in two ways:

a. Numbers between a and b separated by a step size of h

b. A certain number $(k+1)$ of numbers evenly spaced between a and b.

There are two different ways in MATLAB to generate such sequences, depending on how you think of the values in the sequence.

5.2.1 numbers separated by a step size h

To generate a sequence of numbers separated by 1 is done using the `:` symbol, as in `a:b`:

```
>> 1:5                    % 1 2 3 4 5
>> -1:5                   % -1 0 1 2 3 4 5
>> -(1:5)                 % -1 -2 -3 -4 -5
```

The last example shows that the minus sign here means multiply each entry by -1.

If we want a step size of h we use this syntax: `a:h:b`. For instance:

```
>> evens = 0:2:10         % even numbers 0,2,4 ... 10
>> evens = 0:2:9          % even numbers 0,2,4 ... 8. Stops at 8
>> skip15 = 0:15:60       % 0,15,30,45,60 -- class is almost done
>> skip15 = 15*(0:4)      % same thing!
>> skipafew = 1:98        % 1 2 skipafew 97 98
```

Exercise 10:
Find MATLAB commands which generate the following lists. Make sure your answer is correct.

a. The odd numbers 1,3,...99
(32) Answer:

b. The numbers 10,20,30,...120
(33) Answer:

5.2.2 A fixed number of numbers

When plotting functions we will desire a lot (say 100 or a 1000) of evenly-spaced numbers between two points. Rather than figure out the step size between the points it is more convenient to specify how many linearly spaced points we want. This is done with the `linspace` function. Using it as `linspace(a,b)` will produce 100 numbers between a and b. You can override the default of 100 using a third argument:

```
>> linspace(0,5,6)              % ans = [0 1 2 3 4 5] (6 numbers)
>> linspace(0,pi,3)             % ans = [0, pi/2, pi] (3 numbers)
```

Exercise 11:

a. What `linspace` command produces this output:

```
1.000 1.500 2.000 2.500 3.000 3.500 4.000
```

(34) Circle one:
1. `linspace(1,7,4)`
2. `linspace(1,4,1/2)`
3. `linspace(1,4,7)`
4. `linspace(1,1/2,4)`

b. What is the last value output by the command

```
>> linspace(0,pi)
```

(35) Answer: _____

Expression	MATLAB value	Interpreted as
2*3+4*5	26	(2*3)+(4*5)
2+3*4	14	2+(3*4)
(2+3)*4	20	parentheses decides the order
2/5*4	1.6	(2/5)*4
2/(5*4)	0.1	parentheses decides the order
2^3^2	64	(2^3)^2
2^(3^2)	512	parentheses decides the order
5/2/4	0.625	(5/2)/4
5/(2/4)	10	parentheses decides the order

6.3 Basic Functions

Elementary Mathematical functions		
MATLAB notation	Mathematical notation	Meaning of the operation
`sqrt(x)`	$\sqrt{x}$	square root
`abs(x)`	$\lvert x \rvert$	absolute value
`sign(x)`		sign of x ($+1$, -1, or 0)
`exp(x)`	e^x	exponential function
`log(x)`	$\ln x$	natural logarithm
`log10(x)`	$\log_{10} x$	logarithm base 10
`sin(x)`	$\sin x$	sine
`cos(x)`	$\cos x$	cosine
`tan(x)`	$\tan x$	tangent
`asin(x)`	$\sin^{-1} x$	inverse sine
`acos(x)`	$\cos^{-1} x$	inverse cosine
`atan(x)`	$\tan^{-1} x$	inverse tangent

(because exp(1) means e^1 which equals e) this is the special number e

```
>> exp(1)
ans =
    2.7183
```

NOTE: MATLAB may give unexpected results when working with negative numbers, since its default is to assume inputs and outputs are complex numbers. Try evaluating `sqrt(-4)`, `(-8)^(1/3)`, and `log(-5)`, to see what happens. Note that i represents $\sqrt{-1}$.